Towards Trust-Aware Resource Management in Grid Computing Systems

Farag Azzedin and Muthucumaru Maheswaran

TR*Labs* and University of Manitoba
Winnipeg, MB R3T 2N2
Canada
E-mail: {fazzedin, maheswar}@cs.umanitoba.ca

Abstract

Resource management is a central part of a Grid computing system. In a large-scale wide-area system such as the Grid, security is a prime concern. One approach is to be conservative and implement techniques such as sandboxing, encryption, and other access control mechanisms on all elements of the Grid. However, the overhead caused by such a design may negate the advantages of Grid computing. This study examines the integration of the notion of "trust" into resource management such that the allocation process is aware of the security implications. We present a formal definition of trust and discuss a model for incorporating trust into Grid systems. As an example application of the ideas proposed, a resource management algorithm that incorporates trust is presented. The performance of the algorithm is examined via simulations.

1. Introduction

The Grids [FoK99, FoK01] are positioned as systems that scale up to Internet size environments with machines distributed across multiple organizations and administrative domains. The resource management in Grid systems is challenging due to: (a) geographical distribution of resources, (b) resource heterogeneity, (c) autonomously administered Grid domains having their own resource policies and practices, and (d) Grid domains using different access and cost models.

In Grid systems, with distributed ownership for the resources and tasks, it is important to consider *quality of service* (QoS) and security while allocating resources. Integration of QoS into *resource management systems* (RMSs) has been examined by several researchers [FoR00, Mah99]. However, security is implemented as a separate subsystem of the Grid [FoK98b] and the RMS makes the allocation decisions oblivious of the security implications.

We present the following scenarios to motivate our integration of security considerations into resource management. Suppose resource M is part of the Grid and is allocated to a task T. Two major security issues should be considered: (a) protecting the local data in resource M from unauthorized access by components of T and (b) ensuring the integrity and secrecy of T's local data. Some of the techniques that are widely used for providing these features in distributed systems include sand-boxing [ChI00], encryption [Sch96], and other access control and authentication mechanisms. These mechanisms, however, incur additional overhead.

Based on the above scenarios we hypothesize that if the RMS is aware of the security requirements of the resources and tasks it can perform the allocations such that the "security" overhead can be minimized. This is the goal of the *trust-aware resource management system* (TRMS) studied here. The TRMS achieves this goal by allocating resources considering a "trust relationship" between the *resource provider* (RP) and the *resource consumer* (RC). If an RMS maps a resource request strictly according to the trust, then there can be a severe load imbalance in a large-scale wide area system such as the Grid. On the other hand, considering just the load balance or resource-task affinities, as in existing RMSs, causes inefficient overall operation due to the introduction of the overhead caused by enforcing the required level of security.

In Section 2, we define the notions of trust and reputation and outline mechanisms for computing them. A trust model for Grid systems in presented in Section 3. The trust-aware resource management algorithm is presented in Section 4. The performance of the proposed algorithm is examined in Section 5. Related work is briefly discussed in Section 6.

2. Trust and Reputation

2.1. Definition of Trust and Reputation

The notion of trust is a complex subject relating to a *firm belief* in attributes such as reliability, honesty, and competence of the trusted entity. There is a lack of consensus in the literature on the definition of trust and on what constitutes trust management [Mis96, GrS00, AbH00]. The definition of trust that we will use in this paper is as follows:

> *Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time.*

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE

That is, the *firm belief* is a dynamic value and spans over a set of values ranging from *very trustworthy* to *very untrustworthy*. The *trust level* is built on past experiences and given for a specific context. For example, entity y might trust entity x to use its storage resources but not to execute programs using these resources. The *trust level* is specified within a given time because the *trust level* today between two entities is not necessarily the same *trust level* a year ago.

When making trust-based decisions, entities can rely on others for information pertaining to a specific entity. For example, if entity x wants to make a decision of whether to use machine M_j which is unknown to x, then x can rely on the reputation of M_j. The definition of reputation that we will use in this paper is as follows:

> *The reputation of an entity is an expectation of its behavior based on other entities' observations or information about the entity's past behavior at a given time.*

2.2. Computing Trust and Reputation

In computing trust and reputation, several issues have to be considered. First, the trust decays with time. For example, if x trusts y at level p based on past experience five years ago, the trust level today is very likely to be lower unless they have interacted since then. Similar time-based decay also applies for reputation. Second, entities may form alliances and as a result would tend to trust their allies and business partners more than they would trust others. Finally, the *trust level* that x holds about y is based on x's direct relationship with y as well as the reputation of y, i.e., the trust model should compute the eventual trust based on a combination of direct trust and reputation and should be able to weigh the two components differently.

Let D_i and D_j denote two domains of entities. The trust relationship at a given time t between the two domains expressed as $\Gamma(D_i, D_j, t)$ is computed based on the direct relationship at time t between D_i and D_j expressed as $\Theta(D_i, D_j, t)$ as well as the reputation of D_j at time t expressed as $\Omega(D_j, t)$. The weights given to direct and reputation relationships are α and β, respectively. Since the "trustworthiness" of D_j is based more on direct relationship with D_i rather than the reputation of D_j, as far as D_i is concerned, α weighs more than β. Direct relationship is computed as a product of the *trust level* in the direct-trust table (DTT) and the *decay function* ($\Upsilon(t - t_{ij})$), where t is the current time and t_{ij} is the time of the last update or the last transaction between D_i and D_j. The time factor t as explained earlier is very critical because information well-received from an entity five years ago might be ill-received today based on the validity of the information as well as how trustworthy is the entity today. The reputation of D_j is computed as the average of the product of the *trust level* in the reputation-trust table (RTT), the *decay function* ($\Upsilon(t-t_{kj})$), and the relationship factor ($R(D_k, D_j)$) for all domains k. Because reputation is based primarily on what other domains say about a particular domain, we introduced the relationship factor R to prevent cheating via collusions among a group of domains. Hence, R will have a higher value if D_k and D_j are unknown or have no prior relationship among each other and a lower value if D_k and D_j are allies or business partners.

$$\begin{aligned} \Gamma(D_i, D_j, t) &= \alpha \times \Theta(D_i, D_j, t) + \beta \times \Omega(D_j, t) \\ \Theta(D_i, D_j, t) &= DTT(D_i, D_j) \times \Upsilon(t - t_{ij}) \end{aligned}$$

$$\Omega(D_j, t) = \frac{\sum_{k=1}^{n} RTT(D_k, D_j) \times R(D_k, D_j) \times \Upsilon(t - t_{kj})}{\sum_{k=1}^{n}(D_k)}$$

Currently, we are developing a trust management architecture that can evolve and maintain the trust values based on the concepts explained above. The rest of this paper is concerned with using the trust values maintained by such a system to perform efficient resource allocation.

3. A Trust Model for Grid Systems

3.1. Trust Model for Grid Systems

In our model, the overall Grid system is divided into *Grid domains* (GDs). The GDs are autonomous administrative entities consisting of a set of resources and clients managed by a single administrative authority. By organizing a Grid as a collection of GDs, issues such as scalability, site autonomy, and heterogeneity can be easily addressed. In our model, we associate two virtual domains with each GD: (a) a *resource domain* (RD) to signify the resources within the GD and (b) a *client domain* (CD) to signify the clients within the GD. As RDs and CDs are virtual domains mapped onto GDs, some instances of RDs and CDs can map onto the same GD.

An RD has the following attributes that are relevant to the TRMS: (a) ownership, (b) set of *type of activity* (ToA) it supports, and (c) *trust level* (TL) for each ToA. The set of ToAs determine the functionalities provided by the resources that are part of the RD. Some example activities a task can engage at an RD include printing, storing data, and using display services. Associating a TL with each ToA provides the flexibility to selectively open services to clients.

Similarly, the CDs have their own trust attributes relevant to the TRMS. The CD trust attributes include: (a) ownership, (b) ToAs sought, and (c) TLs associated with ToAs. The ToA field indicates the type and number of activities

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE

Table 1. Example of a trust level table.

Client Domains	Resource Domains			
	...	RD_j		
	...	TL_j		
	...	A_1	...	A_k
CD_1	...	TL^1_{1j}	...	TL^k_{1j}
$\vdots$	$\vdots$	$\vdots$		
CD_i	...	TL^1_{ij}	...	TL^k_{ij}

Table 2. Expected trust supplement values.

requested TL	offered TL				
	A	B	C	D	E
A	0	0	0	0	0
B	B - A	0	0	0	0
C	C - A	C - B	0	0	0
D	D - A	D - B	D - C	0	0
E	E - A	E - B	E - C	E - D	0
F	F	F	F	F	F

a client is requesting. The ToAs can be atomic or composed. A client with an atomic ToA requires just one activity whereas a client with a composed ToA requires multiple activities.

Table 1 shows an example trust level table between a set of RDs and CDs. The entries in the trust level table are *symmetric* quantifiers for the trust relationships that are asymmetric. For example, let the trust relationship between client domain CD_i and resource domain RD_j be defined by $f(i,j)$. Because trust is an asymmetric function the reverse relationship between RD_j and CD_i, in general, is not given by $f(i,j)$. However, in Table 1, we denote the current value of the two functions using a single value, i.e., TL^k_{ij} for CD_i and RD_j engaging in activity A_k. The entry TL^k_{ij} in Table 1 denotes the trust value for an activity of a client from CD_i on a resource in RD_j. Suppose we have client X from CD_i wanting to engage in activities A_p, A_q, and A_r on resource Y at RD_j. From Table 1, we can compute the *offered trust level* (OTL), TL^o_{ij} for the composite activity between X and Y, i.e., $\mathrm{TL}^o_{ij} = \min(\mathrm{TL\ for\ A}_p, \mathrm{TL\ for\ A}_q, \mathrm{TL\ for\ A}_r)$. There are two *required trust levels* (RTLs). One from the client side and the other from the resource side. If the OTL is greater than or equal to the maximum of client and resource RTLs, then the activity can proceed with no additional overhead. Otherwise, there will be additional security overhead involved in supplementing the OTL to meet the requirements.

The trust level values used in Table 2 range from *very low trust level* to *very high trust level* corresponding to A *to* E respectively. Table 2 shows the *expected trust supplement* (ETS) for different RTL and OTL values. The ETS values are given by $\mathrm{RTL} - \mathrm{OTL}$. The ETS value is zero, when $\mathrm{RTL} - \mathrm{OTL} < 0$. It can be noted from Table 2 that the RTL has a value F that is not provided by OTL. This is supported in the model so that client or resource domains can enforce enhanced security by increasing their RTL value to F.

A straight forward approach to creating and maintaining the trust level table can result in an inefficient process in a very large-scale system such as the Grid. This process is made efficient in our model by various methods. First, as mentioned previously, we divide the Grid system into GDs. The resources and clients within a GD inherit the parameters associated with the RD and CD that are associated with the GD. This increases the scalability of the overall approach. Second, trust is a slow varying attribute, therefore, the update overhead associated with the trust level table is not significant. A value in the trust level table is modified by a new trust level value that is computed based on a *significant* amount of transactional data.

Figure 1 shows a block diagram of a trust-aware RMS. The CDs and RDs have agents associated with them that monitor the Grid level transactions and form the trust notions. These agents have access to the trust level table. If the new trust values they form are different from the existing values in the tables, the agents update the table. In this study, we maintain a single table in a centrally organized RMS. The table may, however, be replicated at different domains for reading purposes.

As shown in Figure 1, a CD or RD agent can estimate trust via direct and recommender channels. The direct channel is estimating the trust based on direct transactions and the recommender channel is estimating the trust based on reputation. The recommender may be a set of CD or RD agents that had previous interactions with the domain of interest. The target CD or RD agent that receives the recommendation will decide on how to form the eventual trust value using the recommender and direct trusts as input values.

4. Trust-Aware Resource Management System Algorithm

As an example application of the above mentioned trust integration, in this section, we present a *Trust-aware Resource Management* (TRM) algorithm. In this algorithm, clients belonging to different CDs present the requests for task executions. The TRM algorithm allocates the resources. Different requests belonging to the same CD may be mapped onto different RDs. The TRM scheduler is based on the following assumptions: (a) centralized scheduler organization, (b) non preemptive task execution, and (c) indivisible tasks (i.e., a task cannot be distributed over multiple

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE

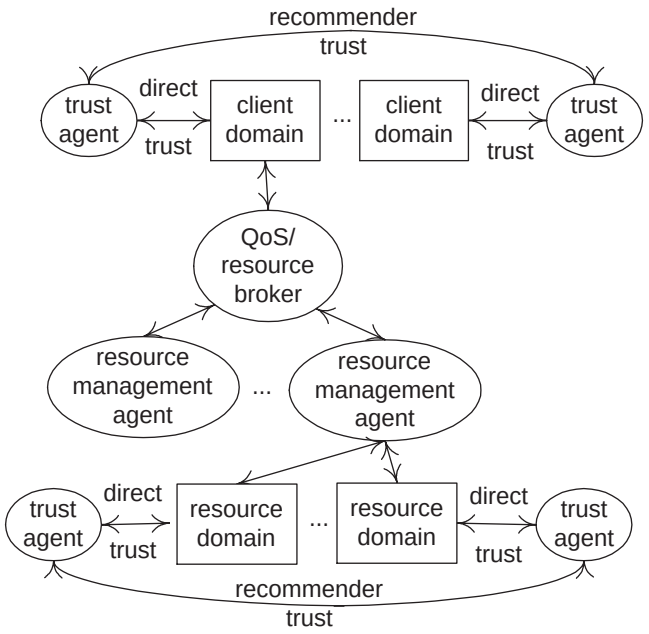

Figure 1. Components of a Grid resource management trust model.

machines).

As shown in the pseudo-code in Figure 2, the TRM algorithm collects client requests for a predefined time interval to form batch of requests, called a *"meta-request"*. The meta-request is then scheduled by the `TRM-schedule` function shown in Figure 3. The `TRM-schedule` function is called when the current time is equal to the `current scheduling event time` that is equal to τ. The `TRM-schedule` function uses a heuristic based on [MaA99] called trust aware min-min heuristic to map the meta-requests.

(1) $\tau = \tau_0$;; scheduler start time
(2) $\Delta\tau$;; inter-schedule time
(3) **while** (true)
(4) $\tau = \tau + \Delta\tau$
(5) **do** until (current time $\geq \tau$)
(6) collect arriving CD requests into meta-request R_a
(7) **enddo**
(8) $R_s = R_a$
(9) TRM-schedule (R_s, $\tau + \Delta\tau$)
(10) some requests in R_s may not have been scheduled – they are inserted back into R_a
(11) $R_a = R_a + R_s$
(12) **endwhile**

Figure 2. The dynamic scheduler used by the RMS.

Let $t(r_i)$ denote the task being executed by request r_i and $c(r_i)$ denote the originating client. Furthermore, let R_i be the i^{th} meta-request and α_i be the available time of machine M_i after executing all requests assigned to it. Further, α_i^j be the available time α_i after executing all requests that belong to meta-request R_j. Also, let EEC($M_i, t(r_j)$) be the *expected execution cost* for $t(r_j)$ on machine M_i and ESC($M_i, t(r_j)$) be the *expected security cost* if $t(r_j)$ is assigned to machine M_i. The ESC value is a function of the *trust cost* (TC) value obtained from ETS (Table 2) and the task under consideration. Finally, let ECC($M_i, t(r_j)$) denotes the *expected completion cost* of $t(r_j)$ on machine M_i which is computed as the EEC of $t(r_j)$ on machine M_i plus the ESC of $t(r_j)$ on machine M_i. The goal of TRM algorithm is to assign $R_i = \{r_0 \ldots r_{n-1}\}$ such that $\{max_m\{\alpha_m^i\}\}$ is minimized $\forall_m$ where n is the number of requests and m is the number of machines.

Figure 3 shows the trust-aware Min-min algorithm used to implement the `TRM-scheduler`. Initially, the ESC matrix is computed. Lines (11) through (13) initializes the ECC table and lines (18) through (20) delete the request scheduled on machine M_i from the meta-request R_v. The task $t(r_j)$ that was successfully assigned to machine M_i is used to update machine M_i available time α_i which in turn is used to compute or update the expected completion cost for all requests yet to be assigned to machime M_i.

5. Simulation Results and Discussions

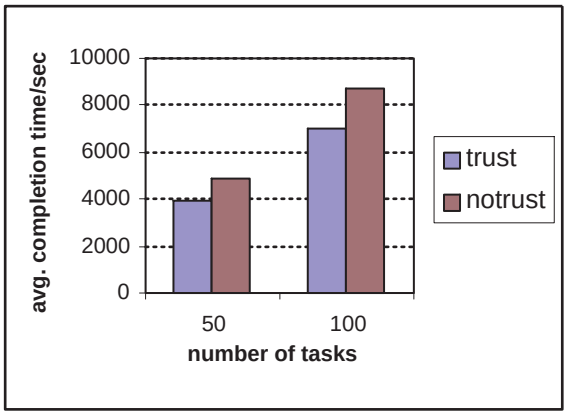

Figure 4. Comparison of average completion time for consistent LoLo heterogeneity.

Simulations were performed to investigate the performance of the trust aware resource management algorithm. The resource allocation process was simulated using a discrete event simulator with request arrivals modeled using a Poisson random process. The number of CDs and RDs were

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE

function TRM-scheduler(meta-request R_v, τ_n)
(1) α_i ;; the available time of machine m_i after executing all requests assigned to it
(2) ETS ;; the ETS values are given by RTL - OTL
(3) $TL(r_j)$;; trust level requested by r_j
(4) OTL ;; is the offered trust level
(5) TC ;; trust cost determined from the ETS table
(6) **for** all machines m_i **do**
(7) **for** all requests r_j in meta-request R_v **do**
(8) OTL = the lowest provided TL among all activities involved in performing $t(r_j)$ on machine m_i
(9) Determine the trust cost from the ETS table
$TC = ETS[TL(r_j), OTL]$
(10) Update the ESC table based on the TC obtained from the ETS table
$ESC[m_i, t(r_j)] = f(TC)$;; ESC resulting value is a function of TC
(11) **for** all r_j in meta-request R_v **do**
(12) **for** all machines m_i **do**
(13) $ECC(m_i, t(r_j))$ = $EEC(m_i, t(r_j))$ + $ESC(m_i, t(r_j))$ + α_i
(14) **do until (** all requests in R_v are scheduled OR the minimum machine completion cost $> \tau_n$ **)**
(15) **for** each request r_k in R_v find the earliest completion cost and the machine that obtains it
(16) Find the request r_j with the minimum earliest completion cost
(17) Assign r_j to the machine m_i that gives the earliest completion cost
(18) Delete task r_j from R_v
(19) Update the vector α_i
(20) Update $ECC(m_i, t(r_j))$ for all j
(21) **enddo**

Figure 3. TRM scheduling algorithm using the trust-aware-Min-min heuristic.

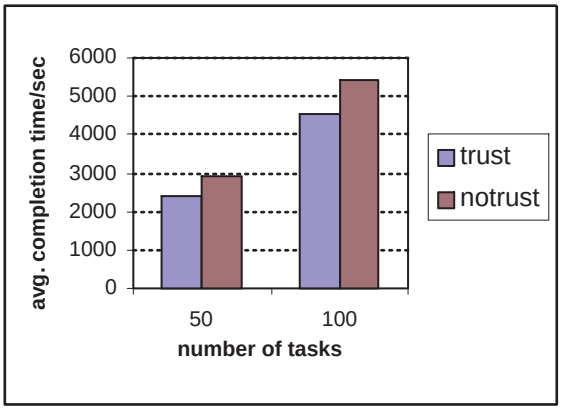

Figure 5. Comparison of average completion time for inconsistent LoLo heterogeneity.

randomly generated from [1-4]. The ToAs required for each request were randomly generated from [1-4] meaning that each $t(r_i)$ involves at least one ToA but no more than 4 ToAs. The two RTL values were randomly generated from [1-6] representing trust levels A to F, respectively. Whereas, the OTL values were randomly generated from [1-5] representing trust levels A to E, respectively.

Two different classes of EEC matrices were used in the simulations. The first class is the consistent *low task and low machine heterogeneity* (LoLo) [MaA99]. This class of EECs model network computing systems that have "related" machines that are "similar" in performance. The tasks that are submitted to the system too have "similar" resource requirements. The second class is the inconsistent LoLo. In this class, the machines are not related.

In the min-min heuristic, the idea is to map a request r_i to machine M_j that gives us the earliest EEC time without considering the security overhead. Although the EEC time was calculated in terms of the execution time of r_i on M_j plus the security overhead of executing r_i on M_j, the security overhead is not considered when mapping r_i to M_j. For the trust aware min-min heuristic, the security overhead is considered while mapping as well as calculating the completion time of executing r_i on M_j.

Figure 4 shows the average completion times of the tasks with five machines for consistent LoLo heterogeneity. From the results it can be observed that if the resource allocator is trust aware, the performance can be improved by about 20%. Figure 5 shows the results from a similar experiment with inconsistent LoLo heterogeneity. The performance improvement in this case was about 13%.

6. Related Work

To the best of our knowledge, no existing literature directly addresses the issue of trust aware resource management. In this section, we examine several papers that examine issues that are peripherally related.

In [FoK98b], a security architecture for a Grid system

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE

is designed and implemented in the context of the Globus system [FoK98]. In [FoK98b], the security policy focuses on authentication and a framework to implement this policy has been proposed.

A design and implementation of a secure *Service Discovery Service* (SDS) is presented in [CzZ99]. SDS can be used by service providers as well as clients. Service providers use SDS to advertise their services that are available or already running while clients use SDS to discover these services.

A model for supporting trust based on experience and reputation is proposed in [AbH00]. This trust-based model allows entities to decide which other entities are trustworthy and also allows entities to tune their understanding of another entity's recommendations.

A survey of trust in Internet applications is presented in [GrS00] and as part of this work a policy specification language called Ponder [DaD01] was developed. Ponder can be used to define authorization and security management policies. Ponder is being extended to allow for more abstract and potentially complex trust relationships between entities across organizational domains.

7. Conclusions

Resource management is a central part of a Grid computing system. In a large-scale wide-area system such a Grid, security is a prime concern. One approach is to be conservative and implement techniques such as sandboxing, encryption, and other access control mechanisms on all elements of the Grid. However, the overhead caused by such a design may reduce the advantages of Grid computing. This study examines the integration of the notion of "trust" into resource management such that the allocation process is aware of the security implications. We present a formal definition of trust and discuss a model for incorporating trust into Grid systems. As an example application of the ideas proposed, a resource management algorithm that incorporates trust is presented. Simulations were performed to evaluate the performance of the resource management algorithm that is trust aware against an algorithm that is trust unaware. The simulation results indicate that the overall performance increases when the resource management algorithm is trust aware.

References

[AbH00] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," *Hawaii Int'l Conference on System Sciences*, 2000.

[ChI00] F. Chang, A. Itzkovitz, and V. Karamcheti, "User-level resource-constrained sandboxing," *4th USENIX Windows Systems Symposium*, Aug. 2000.

[CzZ99] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An architecture for a secure service discovery service," *5th Annual Int'l Conference on Mobile Computing and Networks (MobiCom '99)*, 1999.

[DaD01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder policy specification language," *Workshop on Policies for Distributed Systems and Networks*, 2001.

[FoK01] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *Int'l Journal on Supercomputer Applications*, 2001.

[FoK98] I. Foster and C. Kesselman, "The Globus project: A status report," *7th IEEE Heterogeneous Computing Workshop (HCW '98)*, Mar. 1998, pp. 4–18.

[FoK98b] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational Grids," *ACM Conference on Computers and Security*, 1998, pp. 83–91.

[FoK99] I. Foster and C. Kesselman (eds.), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Fransisco, CA, 1999.

[FoR00] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," *8th Int'l Workshop on Quality of Service (IWQoS '00)*, June 2000.

[GrS00] T. Grandison and M. Sloman, "A survey of trust in Internet applications," *IEEE Communications Surveys & Tutorials*, Vol. 3, No. 4, 2000.

[MaA99] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, Vol. 59, No. 2, Nov. 1999, pp. 107–131.

[Mah99] M. Maheswaran, "Quality of service driven resource management algorithms for network computing," *1999 Int'l Conference on Parallel and Distributed Processing Technologies and Applications (PDPTA '99)*, June 1999, pp. 1090–1096.

[Mis96] B. Misztal, "Trust in modern societies," *Polity Press, Cambridge MA*, Polity Press, Cambridge MA, 1996.

[Sch96] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Second Edition, John Wiley, New York, NY, 1996.

Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)
0-7695-1582-7/02 $17.00 © 2002 IEEE